LIS User's Guide

Submitted under Task Agreement GSFC-CT-2

Cooperative Agreement Notice (CAN) CAN-00OES-01

Increasing Interoperability and Performance of
Grand Challenge Applications in the Earth, Space, Life, and Microgravity
Sciences

September 10, 2004

Revision 3.2

History:

Revision	Summary of Changes	Date
3.2	Milestone "J" submission	September 10, 2004
3.1	Milestone "G" release	July 16, 2004
3.0	Milestone "G" submission	May 7, 2004
2.3	Improvements to Milestone "I"	November 30, 2003
2.2	-	-
2.1	Milestone "I" release	November 10, 2003
2.0	Milestone "I" submission	August 14, 2003
1.1	Milestone "F" release	April 25, 2003
1.0	Milestone "F" submission	March 31, 2003

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland 20771

Contents

1 Introduction

This is the LIS' User's Guide. This document describes how to download and install the code and data needed to run the LIS executable for LIS' "Interoperability Release" milestone – Milestone "J". It describes how to build and run the code, and finally this document also describes how to download output data-sets to use for validation. Updates to this document will provide more detailed instructions on how to configure the executable and will address the graphical user interface.

This document consists of 9 sections, described as follows:

1 **Introduction**: the section you are currently reading

2 **Background**: general information about the LIS project

3 **Preliminaries**: general information, steps, instructions, and definitions used throughout the rest of this document

4 **Running modes**: different parallel running modes of operation

5 **Obtaining the Source Code**: the steps needed to download the source code

6 **Obtaining the Data-sets**: the steps needed to download the data-sets

7 **Building the Executable**: the steps needed to build the LIS executable

8 **Running the Executable**: the steps needed to prepare and submit a run, also describes the various run-time configurations

9 **Output Data Processing**: the steps needed to post-process generated output for visualization

1.1 What's New: LIS 3.0 – LIS-ESMF

This is a proto-type version of LIS using the ESMF. This involves:

1. ESMF components – LIS is broken into two ESMF components, LSM and forcing.

2. ESMF superstructure – An ESMF superstructure manages time and drives the LSM and forcing components.

3. ESMF states – ESMF states are used to encapsulate data that must be shared between the LSM and forcing components. These data are exchanged be means of a "coupler".

4. ESMF configuration utility – The LIS cardfile, *lis.crd*, has been replaced by *lisconfig.sh*, a configuration file based on the ESMF configuration utility.

1.2   What's New: LIS 2.0 – 3.0

1. Running Modes – Now there is more than one way to run LIS. In addition to the standard MPI running mode, there are the GDS running mode and the 1 km running mode. See Section 4 for more details.

2. Sub-domain Selection – Now you are no longer limited to global simulations. You may choose any sub-set of the global domain to run over. See Section 8.2 for more details. (This is currently only available for the MPI-based running mode.)

3. Plug-ins – Now it is easy to add new LSM and forcing data-sets into the LIS driver. See LIS' Developer's Guide for more details.

2 Background

This section provides some general information about the LIS project and land surface modeling.

2.1 LIS

The primary goal of the LIS project is to build a system that is capable of performing high resolution land surface modeling at high performance using scalable computing technologies. The LIS software system consists of a number of components: (1) LIS driver: the core software that integrates the use of land surface models, data management techniques, and high performance computing. (2) community land surface models such as CLM [4], Noah [6], and VIC [7], and (3) Visualization and data management tools such as GrADS [1] -DODS [5] server. One of the important design goals of LIS is to develop an interoperable system to interface and interoperate with land surface modeling community and other earth system models. LIS is designed using an object oriented, component-based style. The adaptable interfaces in LIS can be used by the developers to ease the cost of development and foster rapid prototyping and development of applications. The following sections describe the main components of LIS.

2.2 LIS driver

The core of LIS software system is the LIS driver that controls program execution. The LIS driver is a model control and input/output system (consisting of a number of subroutines, modules written in Fortran 90 source code) that drives multiple offline one-dimensional LSMs. The one-dimensional LSMs such as CLM and Noah, apply the governing equations of the physical processes of the soil-vegetation-snowpack medium. These land surface models aim to characterize the transfer of mass, energy, and momentum between a vegetated surface and the atmosphere. When there are multiple vegetation types inside a grid box, the grid box is further divided into "tiles", with each tile representing a specific vegetation type within the grid box, in order to simulate sub-grid scale variability.

The execution of the LIS driver starts with reading in the user specifications, including the modeling domain, spatial resolution, duration of the run, etc. Section 8 describes the exhaustive list of parameters specified by the user. This is followed by the reading and computing of model parameters. The time loop begins and forcing data is read, time/space interpolation is computed and modified as necessary. Forcing data is used to specify the boundary conditions to the land surface model. The LIS driver applies time/space interpolation to convert the forcing data to the appropriate resolution required by the model. The selected model is run for a vector of "tiles" and output and restart files are written at the specified output interval.

Some of the salient features provided by the LIS driver include:

- Vegetation type-based "tile" or "patch" approach to simulate sub-grid scale variability.

- Makes use of various satellite and ground-based observational systems.

- Derives model parameters from existing topography, vegetation, and soil coverages.

- Extensible interfaces to facilitate incorporation of new land surface models, forcing schemes.

- Uses a modular, object oriented style design that allows "plug and play" of different features by allowing user to select only the components of interest while building the executable.

- Ability to perform regional modeling (only on the domain of interest).

- Provides a number of scalable parallel processing modes of operation.

Please refer to the software design document for a detailed description of the design of LIS driver. The LIS developer's guide describes how to use the extensible interfaces in LIS. The "plug and play" feature of different components is described in this document.

2.3   Community Land Model (CLM)

CLM (Community Land Model) is a 1-D land surface model, written in Fortran 90, developed by a grass-roots collaboration of scientists who have an interest in making a general land model available for public use. LIS currently uses CLM version 2.0. CLM version 2.0 was released in May 2002. The source code for CLM 2.0 is freely available from the National Center for Atmospheric Research (NCAR) [4]. The CLM is used as the land model for the Community Climate System Model (CCSM) (http://www.ccsm.ucar.edu/), which includes the Community Atmosphere Model (CAM) (http://www.cgd.ucar.edu/cms/). CLM is executed with all forcing, parameters, dimensioning, output routines, and coupling performed by an external driver of the user's design (in this case done by LIS). CLM requires pre-processed data such as the land surface type, soil and vegetation parameters, model initialization, and atmospheric boundary conditions as input. The model applies finite-difference spatial discretization methods and a fully implicit time-integration scheme to numerically integrate the governing equations. The model subroutines apply the governing equations of the physical processes of the soil-vegetation-snowpack medium, including the surface energy balance equation, Richards' [12] equation for soil hydraulics, the diffusion equation for soil heat transfer, the energy-mass balance equation for the snowpack, and the Collatz et al. [9] formulation for the conductance of canopy transpiration.

2.4   The Community Noah Land Surface Model

The community Noah Land Surface Model is a stand-alone, uncoupled, 1-D column model freely available at the National Centers for Environmental Prediction (NCEP; [6]). The name is an acronym representing the various developers of the model (N: NCEP; O: Oregon State University, Dept. of Atmospheric Sciences; A: Air Force (both AFWA and AFRL - formerly AFGL, PL); and H: Hydrologic Research Lab - NWS (now Office of Hydrologic Development – OHD)). Noah can be executed in either coupled or uncoupled mode. It has been coupled with the operational NCEP mesoscale Eta model [10] and its companion Eta Data Assimilation System (EDAS) [13], and the NCEP Global Forecast System (GFS) and its companion Global Data Assimilation System (GDAS). When Noah is executed in uncoupled mode, near-surface atmospheric forcing data (e.g., precipitation, radiation, wind speed, temperature, humidity) is required as input. Noah simulates soil moisture (both liquid and frozen), soil temperature, skin temperature, snowpack depth, snowpack water equivalent, canopy water content, and the energy flux and water flux terms of the surface energy balance and surface water balance. The model applies finite-difference spatial discretization methods and a Crank-Nicholson time-integration scheme to numerically integrate the governing equations of the physical processes of the soil vegetation-snowpack medium, including the surface energy balance equation, Richards' [12] equation for soil hydraulics, the diffusion equation for soil heat transfer, the energy-mass balance equation for the snowpack, and the Jarvis [11] equation for the conductance of canopy transpiration.

2.5   Variable Infiltration Capacity (VIC) Model

Variable Infiltration Capacity (VIC) model is a macroscale hydrologic model, written in C, being developed at the University of Washington and Princeton University. The VIC code repository along with the model description and source code documentation is publicly available at the Princeton website [7]. VIC is used in macroscopic land use models such as SEA - BASINS (http://boto.ocean.washington.edu/seasia/intro.htm). VIC is a semi-distributed, grid-based hydrological model, which parameterizes the dominant hydrometeorological processes taking place at the land surface - atmospheric interface. The execution of VIC model requires preprocessed data such as precipitation, temperature, meteorological forcing, soil and vegetation parameters, etc. as input. The model uses three soil layers and one vegetation layer with energy and moisture fluxes exchanged between the layers. The VIC model represents surface and subsurface hydrologic processes on a spatially distributed (grid cell) basis. Partitioning grid cell areas to different vegetation classes can approximate sub-grid scale variation in vegetation characteristics. VIC models the processes governing the flux and storage of water and heat in each cell-sized system of vegetation and soil structure. The water balance portion of VIC is based on three concepts: 1) Division of grid-cell into fraction sub-grid vegetation coverages.

2) The variable infiltration curve for rainfall/runoff partitioning at the land surface.

3) A baseflow/deep soil moisture curve for lateral baseflow.

Water balance calculations are preformed at three soil layers and within a vegetation canopy. An energy balance is calculated at the land surface. A full description of algorithms in VIC can be found in the references listed at the VIC website.

2.6 GrADS-DODS Server

A GrADS-DODS Server (GDS) is a data server built upon the Grid Analysis and Display System (GrADS) and the Distributed Oceanographic Data System (DODS).

GrADS is an earth science data manipulation and visualization tool under development at the Center for Ocean-Land-Atmosphere Studies (COLA) (`http://http://grads.iges.org/cola.html`). See `http://grads.iges.org/grads/grads.html` for more detailed information about GrADS.

DODS, also called the Open source Project for a Network Data Access Protocol (OPeNDAP), is a protocol for serving data-sets stored in various formats over a network. See `http://www.unidata.ucar.edu/packages/dods/` for more detailed information about DODS.

A GDS may be used to provide the LIS driver with the forcing and input parameter data needed to run an LSM.

A GDS is an optional component of the LIS system. LIS may be run without using a GDS to access the forcing and input parameter data-sets. All necessary forcing and input parameter data-sets may be stored on locally-accessable hard-disks and read in directly by the LIS driver, provided the computer system has sufficient memory.

The intent of a GDS for the LIS project is to provide the LIS driver with subsets of the forcing and input parameter data-sets, so that large-scale, high-resolution domains may be broken-up/parallelized and processed across many compute-nodes of a Beowulf cluster.

3 Preliminaries

This code has been compiled and run on both SGI IRIX64 6.5 systems and Linux PC (Intel/AMD based) systems. These instructions expect that you are using such a system. In particular you need

Software:

- SGI

  - MIPSpro version 7.3.1.1m
  - Message Passing Toolkit, mpt, version 1.5.3.0
  - GNU's make, gmake, version 3.77

- Linux

  - Absoft's Pro Fortran Software Developement Kit, version 8.0
    or
    Lahey/Fujitsu's Fortran 95 Compiler, release L6.00c
  - GNU's C and C++ compilers, gcc and g++, version 2.96
  - MPICH , version 1.2.5.2
  - GNU's make, gmake, version 3.77

System Resources:

	1/4 deg	5 km	1 km
memory	250 MB	32 GB	800 MB (per patch)
hard disk space	3 GB	46 GB	850 GB
source	64 MB	64 MB	64 MB
input data	1.5 GB	25 GB	200 GB
output data	1 GB	20 GB	640 GB

Note, the requirements for input and output data are for a 1-day simulation.

You need to create a working directory on your system that has sufficient disk space to install and run in. Throughout the rest of this document this directory shall be refered to as *$WORKING*.

4 Running Modes

The computational and resource requirements increase significantly for global modeling at high resolutions such as 5 km and 1 km. The land surface modeling component in LIS is designed to handle these requirements and perform high-performance, parallel simulation of global, regional, and local land surface processes with a number of land surface models.

LIS is designed to operate in a number of high performance running modes to meet the diverse requirements of distributed memory and shared memory platforms. LIS can operate in two different parallel modes based on the way data is handled by the LIS driver. In the message passing interface (MPI)-based paradigm, a master processor handles data for the entire domain, computes domain decomposition, and subsequently distributes data onto the compute nodes. This paradigm is limited by the amount of memory available to the master processor. On a shared memory platform, a pool of processors can be used to make a large amount of memory available. To handle increased memory requirements and the limited resources available on a distributed memory environment, a GrADS-DODS Server (GDS)-based running mode can be used in LIS. In this mode of operation, the compute nodes retrieve data from a GDS server. This mode of operation is no longer constrained by the lack of a large pool of memory on the master processor.

The LIS driver also includes the capabillity to perform regional modeling in addition to global scales. The domain information can be specified by a user, and the LIS driver handles the subsetting tasks. In the MPI-mode, the subsetting information is derived from a larger domain, whereas in the GDS-mode, the subsetting is carried out by requesting appropriate data from the GDS-server. The details of using these different options are described in the following sections.

Note, *both* parallel running modes require the Message Passing Interface libraries.

4.1 MPI-Based Running Mode

This running mode is not implemented in this proto-type release. LIS must be compiled with MPI support, but it can only be run as a single-process job.

4.2 GDS-Based Running Mode

In order to run LIS using the GDS-based running mode, you must first install a GDS and a DODS enabled version of GrADS. Then you must compile the GDS-based running mode support into LIS' executable.

To install a GrADS-DODS Server simply go to `http://grads.iges.org/grads/gds/` and follow the on-line instructions.

Once you have installed the GDS, you must install a DODS enabled version of GrADS. Go to `http://grads.iges.org/grads/grads.html` and follow the downloading instructions.

After you have installed both the GDS and the GrADS packages, you must edit the GDS data retrieving script, *$WORKING/LIS/opendap_scripts/getdata.pl*. You must modify the definitions of the `$server` and `$GrADS` variables to reflect your installation.

See Section 5 for instructions on downloading the source code, and see Section 7.3 for instructions on compiling the GDS-based running mode support into LIS' executable.

Note, currently, the GDS-based running mode only supports global simulations.

4.3   Non-parallel Running Mode

This running mode is not implemented in this proto-type release. LIS must be compiled with MPI support, but it can only be run as a single-process job.

4.4   1 km Global Runs

The 1 km global run is a special case of the GDS-based running mode. You must first follow the steps in Section 4.2 to properly configure your system. Once configured, you must run the special 1 km scripts.

The main 1 km script drives a pool-of-tasks scheme for parallelizing the computations over the global 1 km domain. This global domain has already been divided into 1183 sub-domain patches, each containing 720×300 gridpoints. The driver script monitors the availability of each compute node of LIS' Linux cluster, and it pushes a sub-domain patch onto each free node. Should, for any reason, the computations on a compute node crash, the corresponding sub-domain patch is returned to the list of patches-to-complete (the "pool"), where it will wait until it is reassigned.

5 Obtaining the Source Code

This section describes how to obtain the source code needed to build the LIS executable.

5.1 Downloading the Source Code

To obtain the source code needed for LIS' "Interoperability Release" revision LIS-ESMF:

1. Go to LIS' "Public Release Home Page"

   Go to http://lis.gsfc.nasa.gov/

   Follow the "Documentation" link.

   Follow the "Milestone J Page"

   Follow the "LIS-ESMF Code Release" link.

2. From LIS' "Public Release Home Page"

   Follow the "LIS-ESMF Source Code and Scripts" link.

3. Download the required *source.tar.gz* and *scripts.tar.gz* files into your working directory, *$WORKING/LIS*.

   Use the "LIS driver and Land Surface Models" link to get *source.tar.gz*.

   Use the "LIS scripts" link to get *scripts.tar.gz*.

4. Unpack these files. Run (in the order listed):

   ```
   % gzip -dc source.tar.gz | tar xf -
   % gzip -dc scripts.tar.gz | tar xf -
   ```

   Unpacking the scripts.tar.gz file will also create the input directory tree needed for downloading the input data-sets.

5. Download the optional (if desired) *postproc.tar.gz* and *opendap_scripts.tar.gz* files into your working directory, *$WORKING/LIS*.

   Use the "Post-processing scripts" link to get *postproc.tar.gz*.

   Use the "GDS scripts" link to get *opendap_scripts.tar.gz*.

   Unpack them.

   ```
   % gzip -dc postproc.tar.gz | tar xf -
   % gzip -dc opendap_scripts.tar.gz | tar xf -
   ```

5.2 Source files

Unpacking the *source.tar.gz* file will create a *$WORKING/LIS/src* sub-directory.
The structure of *src* is as follows:

Directory Name	Synopsis
driver	LIS driver routines
lsm-plugin	Modules defining the function table registry of included LSMs
domains	Routines to create various supported domains
f2lsm	Custom-defined ESMF gridded component coupler
forcingGridComp	Forcing implemented as a gridded ESMF component
lsmGridComp	LSM implemented as a gridded ESMF component
main	ESMF superstructure application driver
domain-plugin	Modules defining function table registries of included domains
forcing-plugin	Modules defining function table registries of included model forcing, observed radiation, and precipitation forcing products
baseforcing	Top level directory for base forcing methods
baseforcing/geos	Routines for handling GEOS forcing product
baseforcing/gdas	Routines for handling GDAS forcing product
interp	Interpolation routines (Based on NCEP's `ipolates` library)
lib	Libraries needed for linking
make	Makefile and needed headers
lsms/clm2	Top level clm2 land surface model sub-directory
lsms/clm2/biogeochem	Biogeochemistry routines
lsms/clm2/biogeophys	Biogeophysics routines (e.g., surface fluxes)
lsms/clm2/camclm_share	Code shared between the clm2 and cam (e.g., calendar information)
lsms/clm2/csm_share	Code shared by all the geophysical model components of the Community Climate System Model (CCSM). Currently contains code for CCSM message passing orbital calculations and system utilities
lsms/clm2/ecosysdyn	Ecosystem dynamics routines (e.g., leaf and stem area index)
clm2/main	Control (driver) routines
clm2/mksrfdata	Routines for generating surface data-sets
clm2/utils	Independent utility routines
lsms/noah.2.6	Noah land surface model version 2.6
lsms/vic	VIC land surface model

Source code documentation may be found on LIS' web-site at
http://lis.gsfc.nasa.gov/Documentation/MilestoneJ/lis/index.html

5.3 Scripts

The *scripts.tar.gz* file contains a script for compiling and building the executable and a sample card file used for running the LIS executable and for configuring the individual runs. These are described in Section 8.

Unpacking the *scripts.tar.gz* file will place the following files into the *$WORKING/LIS* sub-directory:

File Name	Synopsis
lis.crd	Sample card file
comp.csh	Compile and build script
utils	The in-line documentation processing scripts
input	An empty directory tree to hold the input data

5.4 Post-processing

The *postproc.tar.gz* file contains the source and data files needed for generating data-plots using GrADS [1]. Post-processing is described in Section 9.

Unpacking the *postproc.tar.gz* file will create a *$WORKING/LIS/postproc* sub-directory. The structure of *postproc* is as follows:

File Name	Synopsis
noah.25.ctl	GrADS descriptor file for 1/4 deg. Noah data
noah.25.gs	Script for generating 1/4 deg. Noah data plots
clm.25.ctl	GrADS descriptor file for 1/4 deg. CLM data
clm.25.gs	Script for generating 1/4 deg. CLM data plots
vic.25.ctl	GrADS descriptor file for 1/4 deg. VIC data
vic.25.gs	Script for generating 1/4 deg. VIC data plots
pdef	Directory containing global pdef files (only needed when reading 1-d output data files)

5.5 Opendap Scripts

The *opendap_scripts.tar.gz* file contains the scripts used by LIS to access data from a GDS server.

Unpacking the *opendap_scripts.tar.gz* file will create a *$WORKING/LIS/opendap_scripts* sub-directory. The structure of *opendap_scripts* is as follows:

File Name	Synopsis
getdata.pl	Main GDS data-retrieval script
links.sh	Symbolic link generating script

The *getdata.pl* Perl script retrieves data from a GDS by issuing GrADS commands. See the documentation for *getdata.pl* for more details about this script.

Note, you must modify the definitions of the `$server` and `$GrADS` variables to reflect your installation of a GDS and GrADS. See Section 4.2.

LIS never directly calls the *getdata.pl* script by name, rather it calls the script by one of the script's many aliases. These aliases (or symbolic links) must be created before LIS is run. Run the *links.sh* script to generate all the links needed by LIS driver.

6 Obtaining the Data-sets

This section describes how to obtain the data-sets needed to run the LIS executable.

6.1 Downloading the Data-sets

To obtain the data-sets needed for LIS' "Interoperability Release" revision LIS-ESMF:

1. Go to LIS' "Home Page"

   Go to http://lis.gsfc.nasa.gov/

2. Follow the "Get LIS Data" link.

   The Milestone "G" Section provides links to the land surface parameters and atmospheric forcing data.

6.2 Downloading Parameter Data-sets

Land surface models simulate the physical and dynamical processes of the land surface. Driven by external forcing, the spatial and temporal evolution of these processes are intrinsically determined by the physical and dynamical properties, or parameters, of the land surface. Please follow the link to land surface parameters under Milestone "G" to obtain parameter data-sets. It is recommended that the files be organized according to the domain resolution and land surface model type.

   The following section provide examples of how to download the 1/4 deg. parameter data-sets. Downloading the 5 km and 1 km parameter data-sets is similar.

6.2.1 Example: Downloading the 1/4 Deg. Parameter Data-sets via http

To obtain the 1/4 degree parameter data-sets needed for LIS' "Interoperability Release" revision LIS-ESMF using http:

1. From the Milestone "G" Section (See Section 6.1)

   Follow the "Land Surface Parameters" link.

2. Get the "UMD Land/Sea Mask" file from the "Data-sets used by all LIS land surface models" section.

   Use the "HTTP" link to save the UMD land/sea mask file into *$WORKING/LIS/input/GVEG/1_4deg/*

3. Get the "UMD Vegetation Classification map" file from the "Data-sets used by all LIS land surface models" section.

Use the "HTTP" link to save the UMD tile-space vegetation file into
$WORKING/LIS/input/GVEG/1_4deg/

4. Get the "Soil color" file from the "Data-sets used by all LIS land surface models" section.

   Use the "HTTP" link to save the soil color file into
   $WORKING/LIS/input/BCS/1_4deg/

5. Get the "Soil clay fraction" file from the "Data-sets used by all LIS land surface models" section.

   Use the "HTTP" link to save the soil clay fraction file into
   $WORKING/LIS/input/BCS/1_4deg/

6. Get the "Soil sand fraction" file from the "Data-sets used by all LIS land surface models" section.

   Use the "HTTP" link to save the soil sand fraction file into
   $WORKING/LIS/input/BCS/1_4deg/

7. Get the "Canopy height look-up table" from the "CLM Data-sets" section.

   Use the html link to save the canopy height look-up table as *clm2_ptcanhts.txt* into
   $WORKING/LIS/input/BCS/clm_parms

8. Get the "Vegetation classification look-up table" from the "CLM Data-sets" section.

   Use the html link to save the vegetation classification look-up table as *umdvegparam.txt* into
   $WORKING/LIS/input/BCS/clm_parms

9. Get the "Monthly Leaf Area Index" files from the "CLM Data-sets" section.

   Follow the html link.

   Then save the 12 monthly leaf area index files into *$WORKING/LIS/input/AVHRR_LAI*

   Return to main parameter data page.

10. Get the "Monthly Stem Area Index" files from the "CLM Data-sets" section.

    Follow the html link. (The same link as for LAI.)

    Then save the 12 monthly stem area index files into *$WORKING/LIS/input/AVHRR_LAI*

    Return to main parameter data page.

11. Get the "Vegetation look-up table" from the "Noah Data-sets" section.

    Use the html link to save the vegetation look-up table as *noah.vegparms.txt* into
    $WORKING/LIS/input/BCS/noah_parms

12. Get the "Soil look-up table" from the "Noah Data-sets" section.

    Use the html link to save the soil look-up table as *noah.soilparms.txt* into *$WORKING/LIS/input/BCS/noah_parms*

13. Get the "Quarterly albedo climatology" from the "Noah Data-sets" section.

    Follow the html link.

    Then save the 4 quarterly albedo climatology files into *$WORKING/LIS/input/BCS/1_4deg/NOAH*

    Return to main parameter data page.

14. Get the "Monthly greenness fraction climatology" from the "Noah Data-sets" section.

    Follow the html link.

    Then save the 12 monthly greenness fraction climatology files into *$WORKING/LIS/input/BCS/1_4deg/NOAH*

    Return to main parameter data page.

15. Get the "Maximum snow albedo" from the "Noah Data-sets" section.

    Use the "HTTP" link to save the maximum snow albedo files into *$WORKING/LIS/input/BCS/1_4deg/NOAH*

16. Get the "Bottom temperature without elevation correction" files from the "Noah Data-sets" section.

    Use the "HTTP" link to save the bottom temperature without elevation correction file into *$WORKING/LIS/input/BCS/1_4deg/NOAH*

17. Get the "Vegetation look-up table" from the "VIC Data-sets" section.

    Use the html link to save the vegetation look-up table as *veg_lib.txt* into *$WORKING/LIS/input/BCS/vic_parms*

18. Get the "Soil look-up table" from the "VIC Data-sets" section.

    Use the html link to save the soil look-up table as *soil.txt* into *$WORKING/LIS/input/BCS/vic_parms*

Note, files with names ending in *.gz* have been compressed using GNU's gzip. They must be "unzipped" before using. Files with names ending in *.tar.gz* have been packaged using GNU's tar and compressed using GNU's gzip. They must be "unzipped" and "untarred" before using.

6.2.2 Example: Downloading the 1/4 Deg. Parameter Data-sets via GDS

To obtain the 1/4 degree parameter data-sets needed for LIS' "Interoperability Release" revision LIS-ESMF using GDS, consider the following example:

Getting the Noah "Bottom Temperature", **tbot**, data.

Click on the "Tbot" link in the "GDS" column. You will see screen-shot 1.

Then using a DODS enabled GrADS client (see Section 4.2), issue the following GrADS' commands at the GrADS' prompt:

```
ga-> sdfopen http:// -- registered users will see a valid url --
ga-> set fwex
ga-> set t 1
ga-> set x 1 1440
ga-> set y 1 600
ga-> set fwrite -be -sq tbot.bfsa
ga-> set gxout fwrite
ga-> d tbot
ga-> quit
```

where:

"http:// – registered users will see a valid url –" comes from the "DODS URL:" line.

"set t 1" comes from the "Time:" line – (1 points).

"set x 1 1440" comes from the "Longitude:" line – (1440 points, avg. res. $0.25°$).

"set y 1 600" comes from the "Latitude:" line – (600 points, avg. res. $0.25°$).

"d tbot" comes from the list of "Variables:".

This will save the **tbot** data onto your local disk in the file *tbot.bfsa*.

The other data-sets may be obtained from LIS' public GDS in a similar manner. Note, not all the data-sets are in a format suitable for serving through a GDS. Those data-sets must be obtained using the steps outlined in the previous "http" example (Section 6.2.1).

6.3   Downloading the Forcing Data-sets

As mentioned earlier, the land surface models in LIS are forced by model-derived output and satellite and ground-based observations. The data-sets are available through the above link to atmospheric forcing data under the Milestone "G" data page.
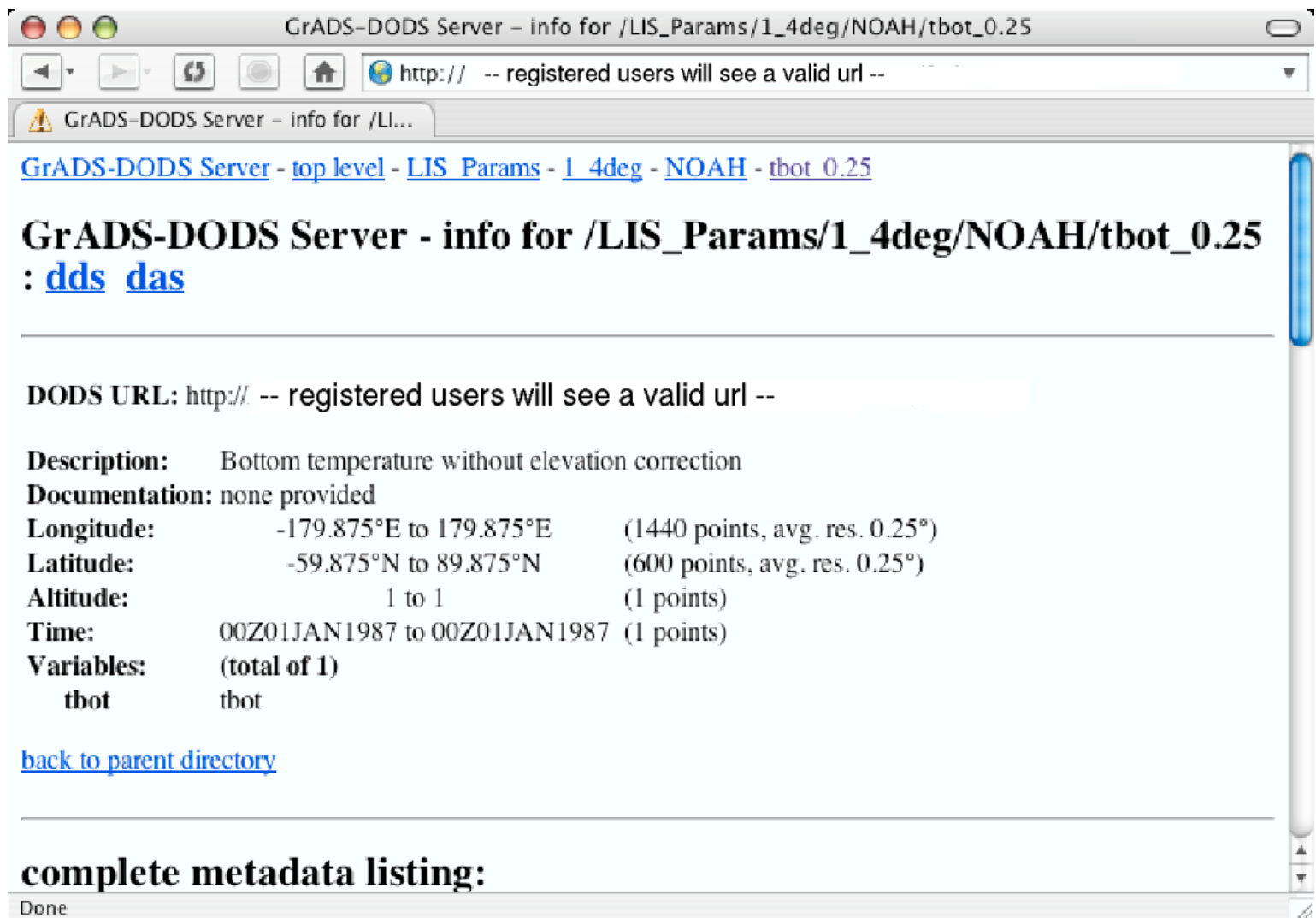
Figure 1: Screen-shot of `tbot` GDS link

6.3.1 Example: Downloading the 1/4 Deg. Forcing Data-sets via http

To obtain the 1/4 degree forcing data-sets needed for LIS' "Interoperability Release" revision LIS-ESMF using http:

1. From the Milestone "G" Section (See Section 6.1)

   Follow the "Atmospheric Forcing Data" link.

2. Get the "GDAS" file.

   Use the "HTTP" link to save the GDAS forcing data-set into
   $WORKING/LIS/input/FORCING/GDAS/

   Note, this is 1-month sample of GDAS forcing data from 01 June 2001 to 30 June 2001.

3. Get the "GEOS3" file.

   Use the "HTTP" link to save the GEOS3 forcing data-set into
   $WORKING/LIS/input/FORCING/GEOS/

   Note, this is 1-month sample of GDAS forcing data from 01 June 2001 to 30 June 2001.

4. Get the "GDAS Elevation Correction" file.

   Follow the html link.

   Use the "HTTP" link to save the GDAS elevation correction data differences file into
   $WORKING/LIS/input/FORCING/

   Return to main forcing data page.

5. Get the "GEOS3 Elevation Correction" file.

   Follow the html link.

   Use the "HTTP" link to save the GEOS3 elevation correction data differences file into
   $WORKING/LIS/input/FORCING/

   Return to main forcing data page.

6. Get the "GEOS4 Elevation Correction" file.

   Follow the html link.

   Use the "HTTP" link to save the GEOS4 elevation correction data differences file into
   $WORKING/LIS/input/FORCING/

   Return to main forcing data page.

7. Get the "AGRMET shortwave flux" file.

   Use the "HTTP" link to save the AGRMET shortwave flux file into *$WORKING/LIS/input/FORCING/AGRMET/*

   Note, this is 1-month sample of AGRMET forcing data from 01 June 2001 to 30 June 2001.

8. Get the "AGRMET longwave flux" file.

   Use the "HTTP" link to save the AGRMET longwave flux file into *$WORKING/LIS/input/FORCING/AGRMET/*

   Note, this is 1-month sample of AGRMET forcing data from 01 June 2001 to 30 June 2001.

9. Get the "CMAP Precipitaion" file.

   Use the "HTTP" link to save the CMAP precipitation file into *$WORKING/LIS/input/FORCING/CMAP/*

   Note, this is 1-month sample of AGRMET forcing data from 01 June 2001 to 30 June 2001.

Note, files with names ending in *.gz* have been compressed using GNU's gzip. They must be "unzipped" before using. Files with names ending in *.tar.gz* have been packaged using GNU's tar and compressed using GNU's gzip. They must be "unzipped" and "untarred" before using.

Note, to obtain data outside of this 1-month sample, you must use the GDS.

Obtaining the forcing data-sets via the GDS is similar to the example given in Section 6.2.2.

6.4 Downloading the Sample Output Data-sets

In addition to the above required parameter and forcing data-sets, LIS provides sample output data-sets that may be downloaded and used for comparision. These data-sets are available through LIS' public data server. You must use a DODS enabled GrADS client (see Section 4.2) to access the data.

6.4.1 Example: Downloading The Sample 1/4 Deg. Output Data-sets Via GDS

To obtain the sample 1/4 degree output data-sets from LIS' "Interoperability Release" revision LIS-ESMF using GDS, consider the following example:

Getting the Noah GEOS output data.

1. Go to LIS' public data server

   Go to http://lisdata.gsfc.nasa.gov:9090/dods/

2. Then follow the "LIS-DATA" → "Output" → "global" → "1_4deg" links.

3. Then click on the "NOAH-GEOS" "info" link.

You will be presented with a page similar to screen-shot 1.

To get only the "net shortwave radiation" variable, issue the following GrADS' commands at the GrADS' prompt:

```
ga-> sdfopen http://http://lisdata.gsfc.nasa.gov:9090/dods/\
LIS-DATA/Output/global/1_4deg/NOAH-GEOS
ga-> set fwex
ga-> set t 1 8
ga-> set x 1 1440
ga-> set y 1 600
ga-> set fwrite -be -sq swnet.bfsa
ga-> set gxout fwrite
ga-> d swnet
ga-> quit
```

To get all variables, issue the following GrADS' commands at the GrADS' prompt:

```
ga-> sdfopen http://http://lisdata.gsfc.nasa.gov:9090/dods/\
LIS-DATA/Output/global/1_4deg/NOAH-GEOS
ga-> set fwex
ga-> set t 1 8
ga-> set x 1 1440
ga-> set y 1 600
ga-> set fwrite -be -sq noah.geos.bfsa
ga-> set gxout fwrite
ga-> d swnet
ga-> d lwnet
ga-> d qle
ga-> d qh
ga-> d qg
ga-> d snowf
ga-> d rainf
ga-> d evap
ga-> d qs
ga-> d qsb
ga-> d qsm
ga-> d delsoilmoist
ga-> d delswe
ga-> d avgsurft
ga-> d albedo
ga-> d swe
ga-> d soilmoist1
ga-> d soilmoist2
ga-> d soilmoist3
ga-> d soilmoist4
```

```
ga-> d soilwet
ga-> d tveg
ga-> d esoil
ga-> d rootmoist
ga-> d wind
ga-> d rainfforc
ga-> d snowfforc
ga-> d tair
ga-> d qair
ga-> d psurf
ga-> d swdown
ga-> d lwdown
ga-> quit
```

See Section 6.2.2 for more details.

Downloading the "CLM-GEOS" and "VIC-GEOS" data-sets is similar.

6.4.2   Viewing The Sample 1/4 Deg. Output Data-sets

Instead of downloading these sample output data-sets, you may view the data on-line. Follow the instructions in Section 6.4.1 that take you to the "NOAH-GEOS" "info" page on LIS' public data server. The follow the "Visualize!" link for whichever variable you wish to see. You will be presented with an easy-to-use interface for examining a data-plot of your chosen variable.

7 Building the Executable

This section describes how to build the source code and create LIS' executable – named LIS.

First perform the steps described in Section 5.

If you are building on a Linux pc system, you must edit the *Makefile* located in *$WORKING/LIS/src/make*. Change the definition of `MPI_PREFIX` to the directory where you installed MPICH. Currently `MPI_PREFIX` is set to */data1/jim/local/mpich-1.2.4-absoft*. Next, you must edit the definition of `ESMF_ARCH`. Currently it is defined to be `linux_absoft`, which specifies that you are using Absoft's Fortran compiler. If you are using Lahey's Fortran compiler, change `ESMF_ARCH` to `linux_lf95`.

Then

1. Change directory into *$WORKING/LIS/*.

    `% cd $WORKING/LIS/`

2. Run the compiling script.

    `% ./comp.csh`

See Appendix B to see the Makefile.

7.1 General Build Instructions

This section describes how to build the LIS code on a platform other than those discussed in Section 3.

7.1.1 Required Software Libraries

In order to build the LIS executable, the following libraries must be installed on your system:

- Message Passing Interface (MPI)

    - vendor supplied, or
    - MPICH
      (http://http://www-unix.mcs.anl.gov/mpi/mpich/)

- Earth System Modeling Framework (ESMF) version 2.0.0rp2 (http://www.esmf.ucar.edu/)

- bacio

- w3lib

To install the MPI libraries, follow the instructions provided at the MPI URL listed above.

Note: Due to the mix of programing languages (Fortran and C) used by LIS, you may run into linking errors when building the LIS executable.

When compiling code using Absoft's Pro Fortran SDK, set the following compiler options:

-YEXT_NAMES=LCS -s -B108 -YCFRL=1

These must be set for each of the above libraries.

7.1.2 Modifying the Makefile

This section lists the variables in the *Makefile* that must be set by the user before compiling.

Variable	Description
UNAMES	set by call to uname to determine what type of system you are using. Determine which variable (UNAMES or UMACHINE) will contain the needed information and use it.
UMACHINE	set by call to uname to determine what type of system you are using. Determine which variable (UNAMES or UMACHINE) will contain the needed information and use it.
MPI_PREFIX	path to where mpi libraries are installed
LIB_MPI	path to mpi libraries
INC_MPI	path to mpi header files
ESMF_DIR	path to where esmf libraries are installed
LIB_ESMF	path to esmf libraries
MOD_ESMF	path to esmf modules
ESMF_ARCH	system on which esmf libraries were compiled
FC	fortran compiler
CPP	C preprocessor
LIB_DIR	path to where lis libraries are installed
CPPFLAGS	flags for C preprocessor
CFLAGS	flags for C compiler
FFLAGS	flags for Fortran compiler
FOPTS	additional options for compiler and linker
LDFLAGS	flags for linker
NEW_ARCH_HERE	replace this with the type of system you are using, either the result from uname -s or uname -m. E.g., IRIX64 or i686

Note: For Linux architectures, the default ESMF_ARCH is set to be linux_absoft. For Linux systems using the Lahey Fortran compiler, ESMF_ARCH must be changed to linux_lf95.

7.2 Compiling non-parallel running mode

This running mode is not implemented in this proto-type release. LIS must be compiled with MPI support, but it can only be run as a single-process job.

7.3 Compiling GrADS-DODS Support

The above building instructions generate a LIS executable that reads input data off of local disk (the MPI-based running mode). To compile an executable that uses a GrADS-DODS server [1] to retrieve input data files (the GDS-based running mode), you must edit the *Makefile*. Find the appropriate `FFLAGS` definition in the *Makefile*. Add `-DOPENDAP` to the end of the definition. Then follow the above building instructions.

7.4 Generating documentation

LIS code uses the ProTex documenting system [2]. The documentation in LaTeX format can be produced by typing `gmake doc` in the *$WORKING/LIS/src/make* directory. This command produces documentation, generating all the files in *$WORKING/LIS/doc* directory. These files can be easily converted to pdf or html formats using utilites such as `pdflatex` or `latex2html`.

---

[1]This LIS distribution is configured to retrieve data using LIS' GrADS-DODS server.

8 Running The Executable

This section describes how to run the LIS executable. Once the LIS executable is built, a simulation can be performed using the *lisconfig.sh* file. As described in Section 4, LIS can only be executed through an *mpirun* script. Assuming that MPI is installed correctly, the LIS simulation is carried out by the following command issued from within the *$WORKING/LIS* directory.

    % mpirun -np 1 ./LIS

The `-np` flag indicates the number of processors used in the run. On a multi-processor machine, the parallel processing capbabilities of LIS can be exploited using this flag.

See Section 8.3 for instructions on running LIS over the global 1 km domain.

8.1 Configuring Run Via LIS Configuration File

This section describes how to configure your LIS run by specifying the options in the *lisconfig.sh* configuration file.

Refer to Appendix A to see a sample lis configuration file. This configuration file is documented. Refer to the comments within it to configure your run.

8.2 Domain Example

This section describes how to compute the values for the `#Domain` section of the *lisconfig.sh* configuration file.

First, we shall generate the values for the parameter data domain. These are the values `Param Lat (LL)`, `Param Lon (LL)`, `Param Lat (UR)`, `Param Lon (UR)`, `Param deltax`, and `Param deltay`. LIS' parameter data is defined on a Latitude/Longitude grid, from -180 to 180 degrees longitude and from -60 to 90 degrees latitude.

Since the parameter data is on a Latitude/Longitude grid, we set

```
Grid Projection: 0
```

For this example, consider running at 1/4 deg resolution. The coordinates of the south-west and the north-east points are specified at the grid-cells' centers. Here the south-west grid-cell is given by the box $(-180, -60), (-179.750, -59.750)$. The center of this box is $(-179.875, -59.875)$. [2]

```
Param Lat (LL): -59.875
Param Lon (LL): -179.875
```

The north-east grid-cell is given by the box $(179.750, 89.750), (180, 90)$. Its center is $(179.875, 89.875)$.

```
Param Lat (UR): 89.875
Param Lon (UR): 179.875
```

---

[2]Note, these coordinates are ordered (longitude, latitude).

Setting the resolution (0.25 deg) gives

```
Param deltax: 0.25
Param deltay: 0.25
```

And this completely defines the parameter data domain.

If you wish to run over the whole domain defined by the parameter data domain then you simply set the values of `Run Lat (LL)`, `Run Lon (LL)`, `Run Lat (UR)`, `Run Lon (UR)`, `Run deltax`, and `Run deltay` equal to the values given by `Param Lat (LL)`, `Param Lon (LL)`, `Param Lat (UR)`, `Param Lon (UR)`, `Param deltax`, and `Param deltay`. This gives

```
Run Lat (LL): -59.875
Run Lon (LL): -179.875
Run Lat (UR): 89.875
Run Lon (UR): 179.875
Run deltax: 0.25
Run deltay: 0.25
```

Now say you wish to run only over the region given by $(-97.6, 27.9), (-92.9, 31.9)$. Since the running domain is a sub-set of the parameter domain, it is also a Latitude/Longitude domain at 1/4 deg. resolution. Thus,

```
Run deltax: 0.25
Run deltay: 0.25
```

Now, since the running domain must fit onto the parameter domain, the desired running region must be expanded from $(-97.6, 27.9), (-92.9, 31.9)$ to $(-97.75, 27.75), (-92.75, 32.0)$. The south-west grid-cell for the running domain is the box $(-97.75, 27.75), (-97.5, 28.0)$. Its center is $(-97.625, 27.875)$; giving

```
Run Lat (LL): 27.875
Run Lon (LL): -97.625
```

The north-east grid-cell for the running domain is the box $(-93, 31.75), (-92.75, 32.0)$. Its center is $(-92.875, 31.875)$; giving

```
Run Lat (UR): 31.875
Run Lon (UR): -92.875
```

This completely defines the running domain.

Note, the LIS project has defined 5 km resolution to be 0.05 deg. and 1 km resolution to be 0.01 deg. If you wish to run at 5 km or 1 km resolution, redo the above example to compute the appropriate grid-cell values.

See Figure 2 for an illustration of adjusting the running grid. See Figures 3 and 4 for an illustration of the south-west and north-east grid-cells.
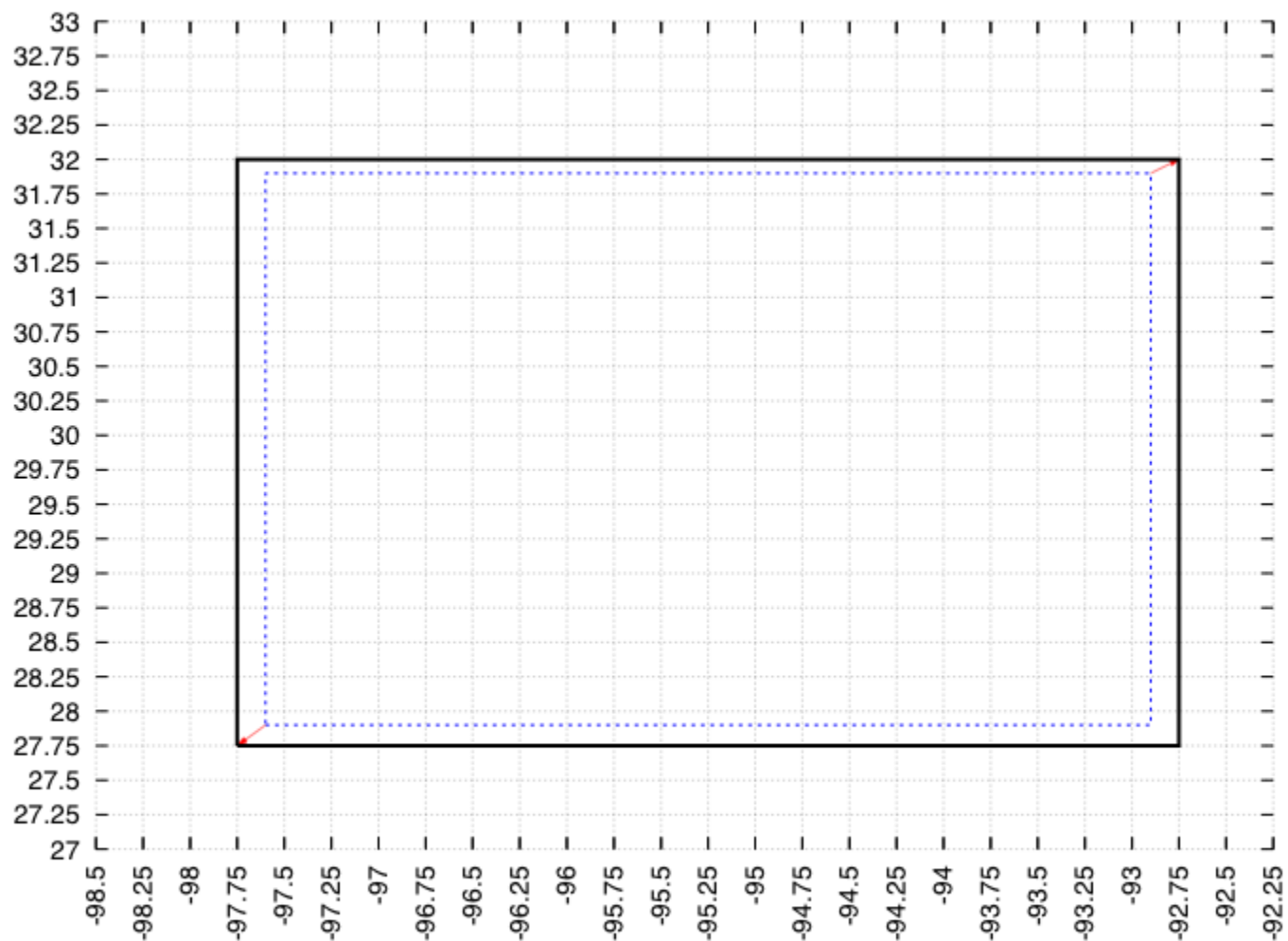
Figure 2: Illustration showing how to fit the desired running grid onto the actual grid
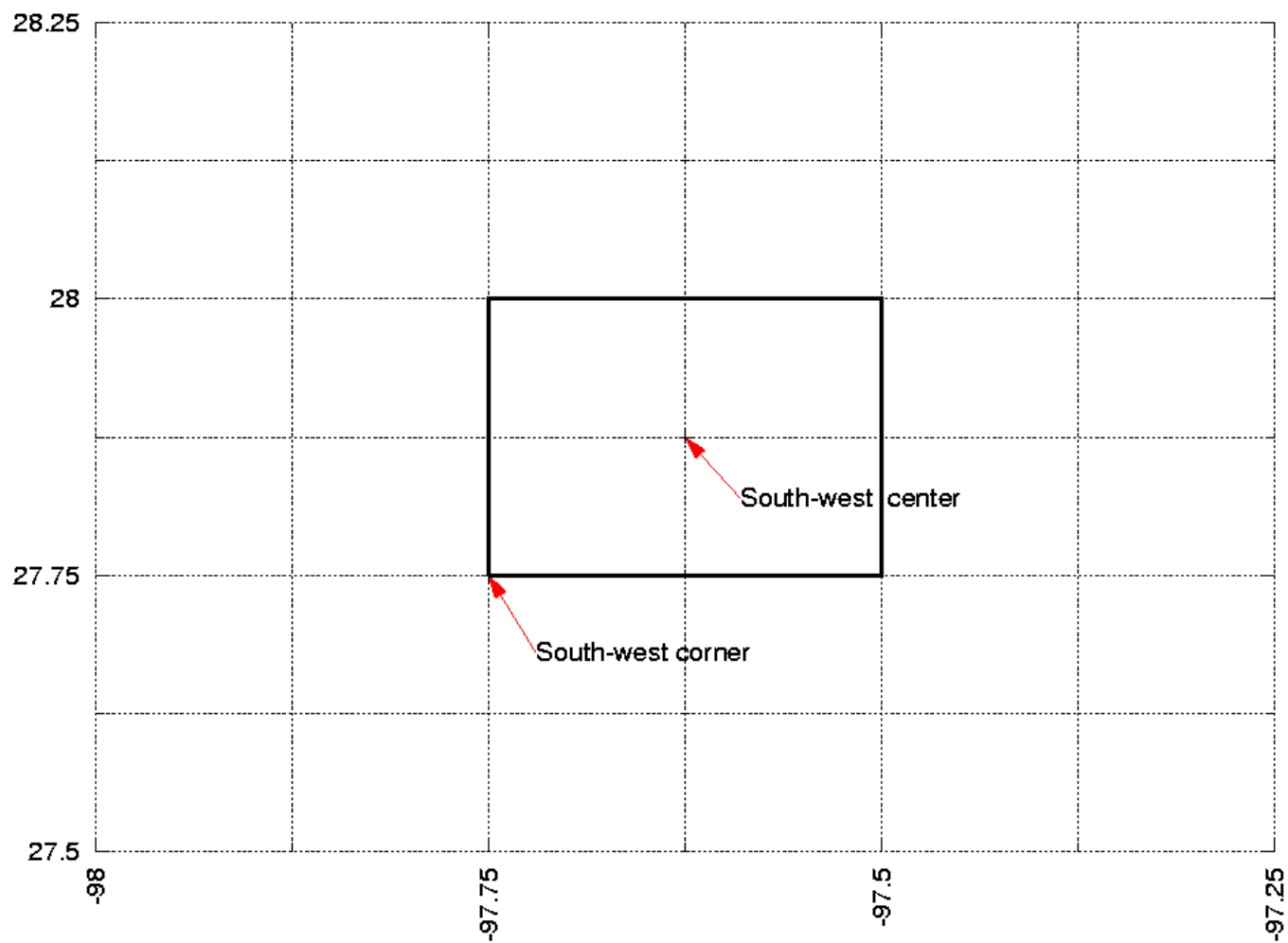
Figure 3: Illustration showing the south-west grid-cell corresponding to the example in Section 8.2
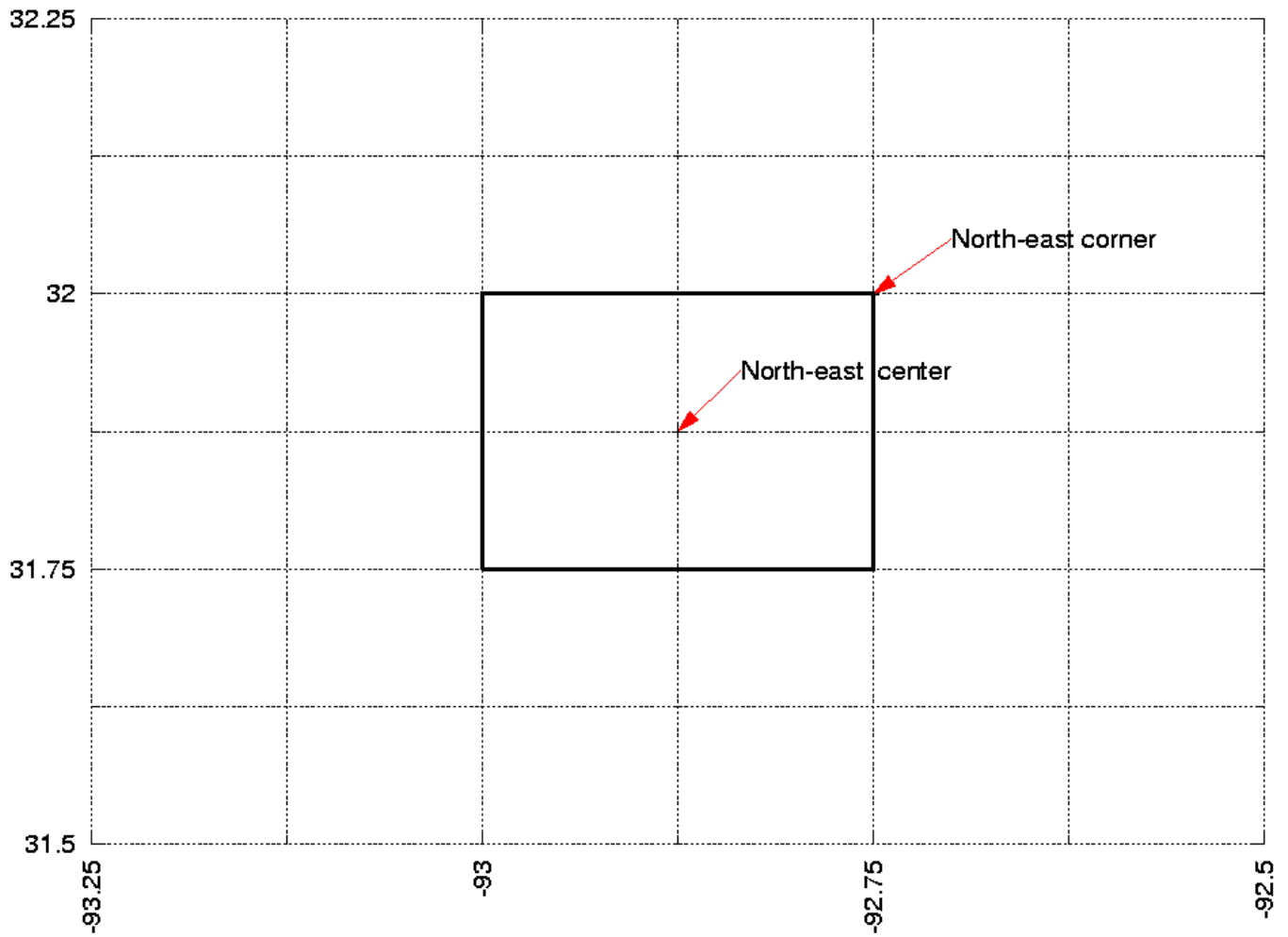
Figure 4: Illustration showing the north-east grid-cell corresponding to the example in Section 8.2

8.3   Running Over The 1 km Domain

Note, these instructions are specific to LIS' Linux cluster.

To run LIS over the global 1 km domain, you must use the special "1 km" scripts.

Before running the scripts you must edit the *farmer.conf* configuration file found in */data1/pool/control*. The variables to set are:

```
cardfile
lsm
exp
```

`cardfile` specifies the full path to the *lisconfig.sh* card file.

`lsm` specifies the land surface model being run. Acceptable values are:

```
NOAH
CLM2
VIC
```

`exp` is a label used to distinguish the simulation. It is used to construct a "snap-shot" directory. This "snap-shot" directory contains information that records which sub-domain patches each compute node processes. This information is used to assemble the output.

After editting the *farmer.conf*, you then run the *create-bones.pl* and *start-farmer-dogs.pl* Perl scripts, which are also in */data1/pool/control*.

```
% cd /data1/pool/control
% ./create-bones.pl
% ./start-farmer-dogs.pl
```

create-bones.pl constructs the "pool" of sub-domains to process. *start-farmer-dogs.pl* starts the simulation.

9 Output Data Processing

This section describes how to process the generated output.

The output data-sets created by running the LIS executable are written into sub-directories of the *$WORKING/LIS/OUTPUT/* directory (created at run-time). These sub-directories are named *EXP999* (by default).

The output data consists of ASCII text files and model output in binary format.

For example, assume that you performed a "1/4 deg Noah with GEOS forcing" simulation for 11 June 2001, with an experiment code value of 999, and writing output as a 2-D array.

This run will produce a *$WORKING/LIS/OUTPUT/EXP999/* directory. This directory will contain:

File Name	Synopsis
Noahstats.dat	Statistical summary of output
NOAH	Directory containing output data

The *NOAH* directory will contain sub-directories of the form *YYYY/YYYYMMDD*, where *YYYY* is a 4-digit year and *YYYYMMDD* is a date written as a 4-digit year, 2-digit month and a 2-digit day; both corresponding to the runnnig dates of the simulation.

For this example, *NOAH* will contain a *2001/20010611* sub-directory.

Its contents are the output files generated by the executable. They are:

2001061100.gs4r

2001061103.gs4r

2001061106.gs4r

2001061109.gs4r

2001061112.gs4r

2001061115.gs4r

2001061118.gs4r

2001061121.gs4r

Note, each file-name contains a date-stamp marking the year, month, day, and hour that the data corresponds to. The output data files for CLM and VIC are similar.

The generated output can be written in a 2-D grid format or as a 1-d vector. See Appendix A for more details. If written as a 1-d vector, the output must be converted into a 2-d grid before it can be visualized. The *postproc.tar.gz*

contains 1-d to 2-d mapping files that GrADS can use for this purpose. [3] See Section 5.4.

In this example, for 2-D output, simply copy the *noah.25.ctl* and *noah.25.gs* files into the *$WORKING/LIS/OUTPUT/EXP999/NOAH/2001/20010611* directory. Then run GrADS:

```
% grads -blc "run noah.25.gs"
```

This will create data-plots of all the variables in *2001061121.gs4r*.

For 1-D output, you must edit the *noah.25.ctl* data descriptor file. Simply uncomment (remove the asterisk) the "PDEF" line. Then run GrADS as above.

---

[3]These mappings only work for "global" runs. If your simulation was over a sub-setted domain, you must either create your own pdef mapping file or simply write 2-d output.

9.1 CLM Output

This table lists the variables written by CLM. The output variables are written to conform to the standards specified by the Assistance for Land-surface Modelling activities (ALMA). See ALMA's web-site [3] for futher details.

ALMA Mandatory Output

Number	Variable	Description	Units
1	SWnet	Net Shortwave Radiation	W/m^2
2	LWnet	Net Longwave Radiation	W/m^2
3	Qle	Latent Heat Flux	W/m^2
4	Qh	Sensible Heat Flux	W/m^2
5	Qg	Ground Heat Flux	W/m^2
6	Snowf	Snowfall rate	kg/m^2/s
7	Rainf	Rainfall rate	kg/m^2/s
8	Evap	Total Evapotranspiration	kg/m^2/s
9	Qs	Surface Runoff	kg/m^2/s
10	Qsb	Subsurface Runoff	kg/m^2/s
11	Qsm	Snowmelt	kg/m^2/s
12	DelSoilMoist	Change in soil moisture	kg/m^2
13	DelSWE	Change in snow water equivalent	kg/m^2
14	SnowT	Snow Temperature	K
15	VegT	Vegetation Canopy Temperature	K
16	BaresoilT	Temperature of bare soil	K
17	AvgSurfT	Average Surface Temperature	K
18	RadT	Surface Radiative Temperature	K
19	Albedo	Surface Albedo, All Wavelengths	-
20	SWE	Snow Water Equivalent	kg/m^2
21	SoilTemp1	Average layer 1 soil temperature	K
22	SoilTemp2	Average layer 2 soil temperature	K
23	SoilTemp3	Average layer 3 soil temperature	K
24	SoilTemp4	Average layer 4 soil temperature	K
25	SoilTemp5	Average layer 5 soil temperature	K

26	SoilTemp6	Average layer 6 soil temperature	K
27	SoilTemp7	Average layer 7 soil temperature	K
28	SoilTemp8	Average layer 8 soil temperature	K
29	SoilTemp9	Average layer 9 soil temperature	K
30	SoilTemp10	Average layer 10 soil temperature	K
31	SoilMoist1	Average layer 1 soil moisture	kg/m^2
32	SoilMoist2	Average layer 2 soil moisture	kg/m^2
33	SoilMoist3	Average layer 3 soil moisture	kg/m^2
34	SoilMoist4	Average layer 4 soil moisture	kg/m^2
35	SoilMoist5	Average layer 5 soil moisture	kg/m^2
36	SoilMoist6	Average layer 6 soil moisture	kg/m^2
37	SoilMoist7	Average layer 7 soil moisture	kg/m^2
38	SoilMoist8	Average layer 8 soil moisture	kg/m^2
39	SoilMoist9	Average layer 9 soil moisture	kg/m^2
40	SoilMoist10	Average layer 10 soil moisture	kg/m^2
41	SoilWet	Total Soil Wetness	-
42	TVeg	Vegetation transpiration	$kg/m^2/s$
43	ESoil	Bare soil evaporation	$kg/m^2/s$
44	RootMoist	Root zone soil moisture	kg/m^2
45	ACond	Aerodynamic conductance	m/s

ALMA Optional Forcing Output

Number	Variable	Description	Units
46	Wind	Near surface wind magnitude	m/s
47	Rainf	Rainfall rate	$kg/m^2/s$
48	Snowf	Snowfall rate	$kg/m^2/s$
49	Tair	Near surface air temperature	K
50	Qair	Near surface specific humidity	kg/kg
51	PSurf	Surface pressure	Pa
52	SWdown	Surface incident shortwave radiation	W/m^2
53	LWdown	Surface incident longwave radiation	W/m^2

9.2 Noah Output

This table lists the variables written by Noah. The output variables are written to conform to the standards specified by the Assistance for Land-surface Modelling activities (ALMA). See ALMA's web-site [3] for futher details.

ALMA Mandatory Output

Number	Variable	Description	Units
1	SWnet	Net Shortwave Radiation	W/m^2
2	LWnet	Net Longwave Radiation	W/m^2
3	Qle	Latent Heat Flux	W/m^2
4	Qh	Sensible Heat Flux	W/m^2
5	Qg	Ground Heat Flux	W/m^2
6	Snowf	Snowfall rate	kg/m^2/s
7	Rainf	Rainfall rate	kg/m^2/s
8	Evap	Total Evapotranspiration	kg/m^2/s
9	Qs	Surface Runoff	kg/m^2/s
10	Qsb	Subsurface Runoff	kg/m^2/s
11	Qsm	Snowmelt	kg/m^2/s
12	DelSoilMoist	Change in soil moisture	kg/m^2
13	DelSWE	Change in snow water equivalent	kg/m^2
14	AvgSurfT	Average Surface Temperature	K
15	Albedo	Surface Albedo, All Wavelengths	-
16	SWE	Snow Water Equivalent	kg/m^2
17	SoilTemp1	Average layer 1 soil temperature	K
18	SoilTemp2	Average layer 2 soil temperature	K
19	SoilTemp3	Average layer 3 soil temperature	K
20	SoilTemp4	Average layer 4 soil temperature	K
21	SoilMoist1	Average layer 1 soil moisture	kg/m^2
22	SoilMoist2	Average layer 2 soil moisture	kg/m^2
23	SoilMoist3	Average layer 3 soil moisture	kg/m^2
24	SoilMoist4	Average layer 4 soil moisture	kg/m^2
25	SoilWet	Total Soil Wetness	-
26	TVeg	Vegetation transpiration	kg/m^2/s
27	ESoil	Bare soil evaporation	kg/m^2/s
28	RootMoist	Root zone soil moisture	kg/m^2

ALMA Optional Forcing Output

Number	Variable	Description	Units
29	Wind	Near surface wind magnitude	m/s
30	Rainf	Rainfall rate	$kg/m^2/s$
31	Snowf	Snowfall rate	$kg/m^2/s$
32	Tair	Near surface air temperature	K
33	Qair	Near surface specific humidity	kg/kg
34	PSurf	Surface pressure	Pa
35	SWdown	Surface incident shortwave radiation	W/m^2
36	LWdown	Surface incident longwave radiation	W/m^2

9.3   VIC Output

This table lists the variables written by VIC. The output variables are written to conform to the standards specified by the Assistance for Land-surface Modelling activities (ALMA). See ALMA's web-site [3] for futher details.

ALMA Mandatory Output

Number	Variable	Description	Units
1	SWnet	Net Shortwave Radiation	W/m^2
2	LWnet	Net Longwave Radiation	W/m^2
3	Qle	Latent Heat Flux	W/m^2
4	Qh	Sensible Heat Flux	W/m^2
5	Qg	Ground Heat Flux	W/m^2
6	Rainf	Rainfall rate	$kg/m^2/s$
7	Snowf	Snowfall rate	$kg/m^2/s$
8	Evap	Total Evapotranspiration	$kg/m^2/s$
9	Qs	Surface Runoff	$kg/m^2/s$
10	Qsb	Subsurface Runoff	$kg/m^2/s$
11	Qfz	Re-freezing of water in the snow	$kg/m^2/s$
12	SnowT	Snow Temperature	K
13	AvgSurfT	Average Surface Temperature	K
14	RadT	Surface Radiative Temperature	K
15	Albedo	Surface Albedo, All Wavelengths	-
16	SoilTemp1	Average layer 1 soil temperature	K
17	SoilTemp2	Average layer 2 soil temperature	K
18	SoilTemp3	Average layer 3 soil temperature	K
19	SoilMoist1	Average layer 1 soil moisture	kg/m^2
20	SoilMoist2	Average layer 2 soil moisture	kg/m^2
21	SoilMoist3	Average layer 3 soil moisture	kg/m^2
22	TVeg	Vegetation transpiration	$kg/m^2/s$
23	ESoil	Bare soil evaporation	$kg/m^2/s$
24	SoilWet	Total Soil Wetness	-
25	RootMoist	Root zone soil moisture	kg/m^2

26	SWE	Snow Water Equivalent	kg/m^2
27	Qsm	Snowmelt	$kg/m^2/s$
28	DelSoilMoist	Change in soil moisture	kg/m^2
29	DelSWE	Change in snow water equivalent	kg/m^2
30	ACond	Aerodynamic conductance	m/s

ALMA Optional Forcing Output

Number	Variable	Description	Units
31	Wind	Near surface wind magnitude	m/s
32	Rainf	Rainfall rate	$kg/m^2/s$
33	Snowf	Snowfall rate	$kg/m^2/s$
34	Tair	Near surface air temperature	K
35	Qair	Near surface specific humidity	kg/kg
36	PSurf	Surface pressure	Pa
37	SWdown	Surface incident shortwave radiation	W/m^2
38	LWdown	Surface incident longwave radiation	W/m^2

A LIS Configuration File

This is a sample LIS configuration file. This configuration file configures LIS to run the Noah land surface model at 1/4 degree resolution using GEOS forcing data. The run starts at 21 hours 10 June 2001 and ends at 21 hours 11 June 2001. It will compute over the region given by the box $(-97.75, 27.75)$, $(-92.75, 32.0)$ degrees longitude/latitude.

```
#overall config options
#Domain resolution (5=1/4deg,6=5km,8=1km)
#Land Surface Model (1=Noah,2=CLM,3=VIC,4=Mosaic,5=HySSIB,6=SSIB)
#Base Forcing scheme (1=GDAS,2=GEOS,3=ECMWF,4=NLDAS,5=GSWP,6=BERG)
#Soil scheme (1=FAO,2=STATSGO)
#LAI scheme (1=AVHRR,2=MODIS)

Domain Resolution: 5
lsm:    1
forcing: 2
soil:   2
lai:    2

#Run control options
#Experiment code (3 digit number)
#Vegetation classification (1=UMD)

Experiment code: 999
Vegetation classification: 1
Number of landcover types: 13
Number of forcing variables: 10

#Use elevation correction (1=yes, 0=No)
#Interpolation Scheme (1=bilinear, 2=bilinear+conservative)

Use elevation correction: 1
Type of interpolation: 1

#Output forcing (1=yes,0=no)
#Write bundled output (1=yes,0=no)
#Write parameter output (1=yes,0=no)
#Write vector output (1=yes,0=no)
#Start mode (1=restart, 2=coldstart)

Write Output: 1
Output forcing: 1
Write bundled output: 1
Write parameter output: 0
```

```
Write vector output: 0
Start mode: 2
End mode: 1

#Time Specifications

sss: 0
smn: 0
shr: 21
sda: 10
smo: 6
syr: 2001

ess: 0
emn: 00
ehr: 21
eda: 11
emo: 6
eyr: 2001

#Timestep

ts: 1800
udef: -9999
odir: "OUTPUT"
dfile: "lisdiag"
Obs precip forcing: 0
Obs radiation forcing: 0
maxt: 1
mina: 0.05

#Domain

$(-97.75,27.75)$, $(-92.75,32.0)$ degrees longitude/latitude.
Grid Projection: 0
Run Lat (LL): 27.875
Run Lon (LL): -97.625
Run Lat (UR): 31.875
Run Lon (UR): -92.875
Run deltax: 0.25
Run deltay: 0.25

Param Lat (LL): -59.875
Param Lon (LL): -179.875
Param Lat (UR): 89.875
Param Lon (UR): 179.875
```

```
Param deltax: 0.25
Param deltay: 0.25

#Common parameters
land mask map:          "GVEG/UMD_60mask.bfsa"
veg classification map: "GVEG/UMD_60veg.bfsa"
sand fraction map:      "BCS/1_4deg/sand60.bfsa"
clay fraction map:      "BCS/1_4deg/clay60.bfsa"
soil color fraction map: "BCS/1_4deg/soicol60.bfsa"
elevation correction map:"GVEG/1_4deg/geos3_diff.1gd4r"
AVHRR LAI map location:  "input/AVHRR_LAI"
MODIS LAI map location:  "input/MODIS_LAI"

#Forcing specifications
#GDAS forcing (forcing=1)

GDAS forcing directory: "input/FORCING/GDAS"
GDAS Number of rows:    256
GDAS Number of columns: 512
GDAS Number of variables: 13

#GEOS forcing (forcing=2)

GEOS forcing directory: "inputFORCING/GEOS/BEST_LK"
GEOS Number of rows:    181
GEOS Number of columns: 360
GEOS Number of variables: 13

#Land Surface Model specifications
#Noah
Noah output frequency (hrs):     3
Noah restart file:              "noah.rst"
Noah greeness fraction directory:"BCS/1_4deg/NOAH/"
Noah quarterly albedo directory: "BCS/1_4deg/NOAH/"
Noah vegetation parameters:     "BCS/noah_parms/noah.vegparms.txt"
Noah soil parameters:           "BCS/noah_parms/noah.soilparms.txt"
Noah max snow free albedo map:   "BCS/1_4deg/NOAH/maxsnalb.bfsa"
Noah bottom temperature map:     "BCS/1_4deg/NOAH/tbot.bfsa"
Noah initial soil moisture (% vol.): 0.30
Noah initial soil temperature (K):   290.0
Noah number of veg parameters:       7
Noah number of soil parameters:      10

#CLM2
CLM2 output frequency (hrs):         3
CLM2 restart file:                  "clm2.rst"
```

```
CLM2 vegetation parameters:          "BCS/clm_parms/umdvegparam.txt"
CLM2 canopy height parameters:       "BCS/clm_parms/clm2_ptcanhts.txt"
CLM2 initial soil moisture (% vol.): 0.45
CLM2 initial soil temperature (K):   290.0
CLM2 initial snow mass (mm H20):       0.

#VIC
VIC output frequency (hrs): 3
VIC Number of Soil Layers: 3
VIC Number of thermal nodes: 5
VIC Number of snowbands: 1
VIC Number of rootzones: 2
VIC soil lookup table:  "BCS/vic_parms/soil.txt"
VIC veg lookup table:   "BCS/vic_parms/veg_lib.txt"
VIC restart file: "restart.dat"
VIC Ds file:      "BCS/1_4deg/VIC/ds.1gd4r"
VIC Dsmax file:  "BCS/1_4deg/VIC/dsmax.1gd4r"
VIC Ws file:      "BCS/1_4deg/VIC/ws.1gd4r"
VIC Infilt file: "BCS/1_4deg/VIC/infilt.1gd4r"
VIC depth1 file: "BCS/1_4deg/VIC/depth1.1gd4r"
VIC depth2 file: "BCS/1_4deg/VIC/depth2.1gd4r"
VIC depth3 file: "BCS/1_4deg/VIC/depth3.1gd4r"

OPENDAP data prefix: /home/lis
```

B Makefile

```
# Set up special characters
null :=
space := $(null) $(null)
doctool :=../../utils/docsgen.sh

# Check for directory in which to put executable
ifeq ($(MODEL_EXEDIR),$(null))
MODEL_EXEDIR := .
endif

# Check for name of executable
ifeq ($(EXENAME),$(null))
EXENAME := LIS
endif

# Check if SPMD is defined in "misc.h"
# Ensure that it is defined and not just "undef SPMD" set in file
ifeq ($(SPMD),$(null))
  SPMDSET := $(shell /bin/grep SPMD misc.h)
  ifneq (,$(findstring define,$(SPMDSET)))
    SPMD := TRUE
  else
    SPMD := FALSE
  endif
endif

# Determine platform
UNAMES := $(shell uname -s)
UMACHINE := $(shell uname -m)

ifeq ($(UNAMES),IRIX64)
ESMF_DIR   := ../lib/esmf
LIB_ESMF   := $(ESMF_DIR)/lib/libO
MOD_ESMF   := $(ESMF_DIR)/mod/modO

endif

ifeq ($(UNAMES),OSF1)

LIB_MPI := /usr/lib
INC_MPI := /usr/include
ESMF_DIR   := ../lib/esmf
LIB_ESMF := $(ESMF_DIR)/lib/libO
MOD_ESMF := $(ESMF_DIR)/mod/modO
```

```
endif

ifeq ($(UMACHINE), i686)

INC_NETCDF := /your/installation/goes/here
LIB_NETCDF := /your/installation/goes/here
MPI_PREFIX := /your/installation/goes/here
LIB_MPI    := $(MPI_PREFIX)/lib
INC_MPI    := $(MPI_PREFIX)/include
ESMF_DIR   := ../lib/esmf
LIB_ESMF   := $(ESMF_DIR)/lib/libO
MOD_ESMF   := $(ESMF_DIR)/mod/modO

endif

# Load dependency search path.
dirs := . $(shell cat Filepath)
# Set cpp search path, include netcdf
cpp_dirs := $(dirs) $(INC_NETCDF) $(INC_MPI)
cpp_path := $(foreach dir,$(cpp_dirs),-I$(dir)) # format for command line

# Expand any tildes in directory names. Change spaces to colons.
VPATH    := $(foreach dir,$(cpp_dirs),$(wildcard $(dir)))
VPATH    := $(subst $(space),:,$(VPATH))

#------------------------------------------------------------------------
# Primary target: build the model
#------------------------------------------------------------------------
all: $(MODEL_EXEDIR)/$(EXENAME)

# Get list of files and determine objects and dependency files
FIND_FILES = $(wildcard $(dir)/*.F $(dir)/*.f $(dir)/*.F90 $(dir)/*.c)
FILES      = $(foreach dir, $(dirs),$(FIND_FILES))
SOURCES   := $(sort $(notdir $(FILES)))
DEPS      := $(addsuffix .d, $(basename $(SOURCES)))
OBJS      := $(addsuffix .o, $(basename $(SOURCES)))
DOCS      := $(addsuffix .tex, $(basename $(SOURCES)))

$(MODEL_EXEDIR)/$(EXENAME): $(OBJS)
 $(FC) -o $@ $(OBJS) $(FOPTS) $(LDFLAGS)
debug: $(OBJS)
        echo "FFLAGS: $(FFLAGS)"
        echo "LDFLAGS: $(LDFLAGS)"
        echo "OBJS: $(OBJS)"
```

```
#************************************************************************
#*********** Architecture-specific flags and rules**********************
#************************************************************************


#-----------------------------------------------------------------------
# SGI
#-----------------------------------------------------------------------


ifeq ($(UNAMES),IRIX64)

ESMF_ARCH = IRIX64
FC          := f90
CPP         := /lib/cpp

# Library directories
LIB_DIR  = ../lib/sgi-64/
HDFLIBDIR  = $(LIB_DIR)
GFIOLIBDIR = $(LIB_DIR)
CPPFLAGS    := -P
PSASINC     :=
CFLAGS      := $(cpp_path) -64 -c -O2 -OPT:Olimit=0 -static -DIRIX64
#CFLAGS      := $(cpp_path) -64 -c -O2 -OPT:Olimit=0 -static -DIRIX64 -DOPENDAP
#CFLAGS      := $(cpp_path) -64 -c -g -static -DIRIX64
FFLAGS      = $(cpp_path) -64 -r4 -i4 -c -cpp -I$(MOD_ESMF)/$(ESMF_ARCH) \
              -DHIDE_SHR_MSG -DNO_SHR_VMATH -DIRIX64 -O2 -OPT:Olimit=0 -static
#FFLAGS      = $(cpp_path) -64 -r4 -i4 -c -cpp -I$(MOD_ESMF)/$(ESMF_ARCH) \
              -DHIDE_SHR_MSG -DNO_SHR_VMATH -DIRIX64 -O2 -OPT:Olimit=0 \
              -static -DOPENDAP
#FFLAGS      = $(cpp_path) -64 -r4 -i4 -c -cpp -I$(MOD_ESMF)/$(ESMF_ARCH) \
              -DHIDE_SHR_MSG -DNO_SHR_VMATH -DIRIX64 -g -static
#FFLAGS      = $(cpp_path) -64 -r4 -i4 -c -cpp -I$(MOD_ESMF)/$(ESMF_ARCH) \
              -DHIDE_SHR_MSG -DNO_SHR_VMATH -DIRIX64 -cif -g -static
FOPTS = $(LIB_DIR)bacio_64_sgi $(LIB_DIR)w3lib_64_sgi
#LDFLAGS      = -L$(LIB_NETCDF) -lnetcdf -L$(LIB_ESMF)/$(ESMF_ARCH) \
              -loldworld -lmpi
#LDFLAGS      = -L$(LIB_ESMF)/$(ESMF_ARCH) -loldworld -lmpi
LDFLAGS      = -64 -L$(LIB_ESMF)/$(ESMF_ARCH) -lesmf -lmpi
# WARNING: -mp and -g together cause wrong answers

# WARNING: - Don't run hybrid on SGI (that's what the -= -mp is all about)

ifeq ($(SPMD),TRUE)
  FFLAGS  -= -mp
  FFLAGS  += -macro_expand
#  FFLAGS  += -I$(INC_MPI) -macro_expand
```

```
#  LDFLAGS += -L$(LIB_MPI) -lmpi
else
  FFLAGS   += -DHIDE_MPI
endif

.SUFFIXES:
.SUFFIXES: .F90 .c .o

.F90.o:
$(FC) $(FFLAGS) $<
.c.o:
cc $(cpp_path) $(CFLAGS) $<

endif
#----------------------------------------------------------------------
# Compaq alpha - Halem cluster
#----------------------------------------------------------------------
ifeq ($(UNAMES),OSF1)

ESMF_ARCH = alpha
FC          := f90
CPP         := /lib/cpp

# Library directories
LIB_DIR  = ../lib/alpha-32/
CPPFLAGS     := -P
PSASINC      :=
CFLAGS       := $(cpp_path) -n32 -DOSF1
FFLAGS       = $(cpp_path) -c -cpp -automatic -convert big_endian \
               -assume byterecl -arch ev6 -tune ev6 -fpe3 \
               -I$(MOD_ESMF)/$(ESMF_ARCH) -DOSF1 \
               -DHIDE_SHR_MSG -DNO_SHR_VMATH
FFLAGS_DOTF90 = -DHIDE_SHR_MSG -DOSF1 -free -fpe3 -DNO_SHR_VMATH
FFLAGS_DOTF = -extend_source -omp -automatic
FOPTS =  $(LIB_DIR)bacio_32_alpha $(LIB_DIR)w3lib_32_alpha
LDFLAGS      = -L$(LIB_ESMF)/$(ESMF_ARCH) -lesmf -lmpi
# WARNING: -mp and -g together cause wrong answers

# WARNING: - Don't run hybrid on SGI (that's what the -= -mp is all about)

ifeq ($(SPMD),TRUE)
  FFLAGS  -= -mp
  FFLAGS  += -I$(INC_MPI) -macro_expand

#  LDFLAGS += -L$(LIB_MPI) -lmpi
```

```
else
  FFLAGS  += -DHIDE_MPI
endif

.SUFFIXES:
.SUFFIXES: .f .f90 .F90 .c .o

.f.o:
$(FC) $(FFLAGS) $<
.F90.o:
$(FC) $(FFLAGS) $<
.c.o:
cc -c $(cpp_path) $(CFLAGS) $<
.f90.o:
$(FC) $(FFLAGS) $<

endif
#----------------------------------------------------------------------
# Linux
#----------------------------------------------------------------------


ifeq ($(UMACHINE),i686)
ESMF_ARCH = linux_absoft

ifeq ($(ESMF_ARCH),linux_ifc)

FC         := $(MPI_PREFIX)/bin/mpif90
CPP        := /lib/cpp

CFLAGS     := $(cpp_path) -c -O2
FFLAGS     = $(cpp_path) -c -I$(MOD_ESMF)/$(ESMF_ARCH) -DHIDE_SHR_MSG \
             -DNO_SHR_VMATH -O
LDFLAGS    = -L$(LIB_ESMF)/$(ESMF_ARCH) -lesmf -lmpich

endif

ifeq ($(ESMF_ARCH),linux_absoft)

FC         := $(MPI_PREFIX)/bin/mpif90
CC         := $(MPI_PREFIX)/bin/mpicc
CPP        := /lib/cpp

# Non opendap
CFLAGS     := $(cpp_path) -c -O2 -DABSOFT -DLITTLE_ENDIAN
FFLAGS     := $(cpp_path) -c -O2 -YEXT_NAMES=LCS -s -B108 -YCFRL=1 \
```

52

```
                       -YDEALLOC=ALL -p$(MOD_ESMF)/$(ESMF_ARCH) -DHIDE_SHR_MSG \
                       -DNO_SHR_VMATH -DABSOFT -DLITTLE_ENDIAN

       # opendap
       #CFLAGS        := $(cpp_path) -c -O2 -DABSOFT -DLITTLE_ENDIAN -DOPENDAP
       #FFLAGS        := $(cpp_path) -c -O2 -YEXT_NAMES=LCS -s -B108 -YCFRL=1 \
                        -YDEALLOC=ALL -p$(MOD_ESMF)/$(ESMF_ARCH) -DHIDE_SHR_MSG \
                        -DNO_SHR_VMATH -DABSOFT -DLITTLE_ENDIAN -DOPENDAP

       # debugging with opendap
       #CFLAGS        := $(cpp_path) -c -g -DABSOFT -DLITTLE_ENDIAN -DOPENDAP
       #FFLAGS        := $(cpp_path) -c -O1 -g -Rb -Rc -Rs -Rp -YEXT_NAMES=LCS -s -B108 \
                        -YCFRL=1 -YDEALLOC=ALL -p$(MOD_ESMF)/$(ESMF_ARCH) \
                        -DHIDE_SHR_MSG -DNO_SHR_VMATH -DABSOFT -DLITTLE_ENDIAN -DOPENDAP

       # debugging
       #CFLAGS        := $(cpp_path) -c -g -DABSOFT -DLITTLE_ENDIAN
       #FFLAGS        := $(cpp_path) -c -O1 -g -Rb -Rc -Rs -Rp -YEXT_NAMES=LCS -s -B108 \
                        -YCFRL=1 -YDEALLOC=ALL -p$(MOD_ESMF)/$(ESMF_ARCH) \
                        -DHIDE_SHR_MSG -DNO_SHR_VMATH -DABSOFT -DLITTLE_ENDIAN

       # profiling -- don't forget the -P in LDFLAGS
       #CFLAGS        := $(cpp_path) -c -O2 -DABSOFT -DLITTLE_ENDIAN -DOPENDAP -pg
       #FFLAGS        := $(cpp_path) -c -O2 -YEXT_NAMES=LCS -s -B108 -YCFRL=1 \
                        -YDEALLOC=ALL -p$(MOD_ESMF)/$(ESMF_ARCH) -DHIDE_SHR_MSG \
                        -DNO_SHR_VMATH -DABSOFT -DLITTLE_ENDIAN -DOPENDAP -P

       LDFLAGS        := -L$(LIB_ESMF)/$(ESMF_ARCH) -L$(LIB_NETCDF) -lnetcdf -lesmf \
                        -lmpich -lU77 -lm

       endif

       ifeq ($(ESMF_ARCH),linux_lf95)

       FC            := $(MPI_PREFIX)/bin/mpif90
       CPP           := /lib/cpp
       CFLAGS        := $(cpp_path) -c -O -DUSE_GCC -DLAHEY -DLITTLE_ENDIAN
       FFLAGS        := $(cpp_path) -c -O -DHIDE_SHR_MSG -DLINUX -DNO_SHR_VMATH \
                        -I$(MOD_ESMF)/$(ESMF_ARCH) -DLAHEY -DLITTLE_ENDIAN
       LDFLAGS       := -L$(LIB_ESMF)/$(ESMF_ARCH) -lesmf -L$(LIB_MPI) -lmpich -s \
                        --staticlink

       endif


       # Library directories
```

```
LIB_DIR  = ../lib/pc-32/$(ESMF_ARCH)/
HDFLIBDIR = $(LIB_DIR)
GFIOLIBDIR = $(LIB_DIR)
CPPFLAGS    := -P
PSASINC     :=
FOPTS = $(LIB_DIR)bacio_32_pclinux $(LIB_DIR)w3lib_32_pclinux
# WARNING: -mp and -g together cause wrong answers

# WARNING: - Don't run hybrid on SGI (that's what the -= -mp is all about)

ifeq ($(SPMD),TRUE)
#  FFLAGS  -= -mp
#  FFLAGS  += -macro_expand
#  FFLAGS  += -I$(INC_MPI) -macro_expand

#  LDFLAGS += -L$(LIB_MPI) -lmpi
else
  FFLAGS  += -DHIDE_MPI
endif

.SUFFIXES:
.SUFFIXES: .F90 .c .o

.F90.o:
$(FC) $(FFLAGS) $<
.c.o:
$(CC) $(cpp_path) $(CFLAGS) $<

endif

RM := rm
# Add user defined compiler flags if set, and replace FC if USER option set.
FFLAGS  += $(USER_FFLAGS)
ifneq ($(USER_FC),$(null))
FC := $(USER_FC)
endif

clean:
$(RM) -f *.o *.mod *.stb  $(MODEL_EXEDIR)/$(EXENAME)

realclean:
$(RM) -f *.o *.d *.mod *.stb  $(MODEL_EXEDIR)/$(EXENAME)
doc:
$(doctool)
#------------------------------------------------------------------------
#!!!!!!!!!!!!!!!!!DO NOT EDIT BELOW THIS LINE.!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
#----------------------------------------------------------------------
# These rules cause a dependency file to be generated for each source
# file.  It is assumed that the tool "makdep" (provided with this
# distribution in clm2/tools/makdep) has been built and is available in
# the user's $PATH.  Files contained in the clm2 distribution are the
# only files which are considered in generating each dependency.  The
# following filters are applied to exclude any files which are not in
# the distribution (e.g. system header files like stdio.h).
#
#  1) Remove full paths from dependencies. This means gnumake will not break
#     if new versions of files are created in the directory hierarchy
#     specified by VPATH.
#
#  2) Because of 1) above, remove any file dependencies for files not in the
#     clm2 source distribution.
#
# Finally, add the dependency file as a target of the dependency rules.  This
# is done so that the dependency file will automatically be regenerated
# when necessary.
#
#     i.e. change rule
#       make.o : make.c make.h
#       to:
#       make.o make.d : make.c make.h
#----------------------------------------------------------------------
DEPGEN := ./MAKDEP/makdep -s F
%.d : %.c
@echo "Building dependency file $@"
@$(DEPGEN) -f $(cpp_path) $< > $@
%.d : %.f
@echo "Building dependency file $@"
@$(DEPGEN) -f $(cpp_path) $< > $@
%.d : %.F90
@echo "Building dependency file $@"
@$(DEPGEN) -f $(cpp_path) $<  > $@
%.d : %.F
@echo "Building dependency file $@"
@$(DEPGEN) -f $(cpp_path) $< > $@
#
# if goal is clean or realclean then don't include .d files
# without this is a hack, missing dependency files will be created
# and then deleted as part of the cleaning process
#
INCLUDE_DEPS=TRUE
ifeq ($(MAKECMDGOALS), realclean)
 INCLUDE_DEPS=FALSE
```

```
endif
ifeq ($(MAKECMDGOALS), clean)
 INCLUDE_DEPS=FALSE
endif

ifeq ($(INCLUDE_DEPS), TRUE)
-include $(DEPS)
endif
```

References

[1] GrADS. http://grads.iges.org/grads/grads.html.

[2] Protex documenting system. http://gmao.gsfc.nasa.gov/software/protex.

[3] ALMA. http://www.lmd.jussieu.fr/ALMA/.

[4] CLM. http://www.cgd.ucar.edu/tss/clm.

[5] DODS. http://www.unidata.ucar.edu/packages/dods/.

[6] Noah. ftp://ftp.ncep.noaa.gov/pub/gcp/ldas/noahlsm/.

[7] VIC. http://hydrology.princeton.edu/research/lis/index.html.

[8] W3FI63 program. http://dss.ucar.edu/datasets/ds609.1/software/mords/w3fi63.f.

[9] G. J. Collatz, C Grivet, J. T. Ball, and J. A. Berry. Physiological and environmental regulation of stomatal conducatance: Photosynthesis and transpiration: A model that includes a laminar boundary layer. *Agric. For. Meteorol.*, 5:107–136, 1991.

[10] Chen. F., Mitchell. K., Schaake. J, Xue. J, Pan. H, Koren. V., Ek. M Duan, and A. Betts. Modeling of land-surface evaporation by four schemes and comparison with fife observations. *J. Geophys. Res.*, 101(D3):7251–7268, 1996.

[11] P. G. Jarvis. The interpretation of leaf water potential and stomatal conductance found in canopies of the field. *Phil. Trans. R. Soc.*, 273:593–610, 1976.

[12] L. A. Richards. Capillary conduction of liquids in porous media. *Physics*, 1:318–333, 1931.

[13] E. Rogers, T. L. Black, D. G. Deaven, G. J. DiMego, Q. Zhao, M. Baldwin, N. W. Junker, and Y. Lin. Changes to the operational "early" eta analysis/forecast system at the national centers of environmental prediction. *Wea. Forecasting*, 11:391–413, 1996.